Published before August 1999
Terms used **lazy access object pointer memory reference**

Found **399** of **99,278**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Special issue on persistent object systems: Adaptable pointer swizzling strategies in object bases: design, realization, and quantitative analysis
Alfons Kemper, Donald Kossmann
July 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 4 Issue 3
Full text available: pdf(2.69 MB) Additional Information: full citation, abstract, references, citings

In this article, different techniques for *"pointer swizzling"* are classified and evaluated for optimizing the access to main-memory resident persistent objects. To speed up the access along inter-object references, the persistent pointers in the form of unique object identifiers (OIDs) are transformed (swizzled) into main-memory pointers (addresses). Pointer swizzling techniques can be divided into two classes: (1) those that allow replacement of swizzled objects from the buffer before th ...

Keywords: object-oriented database systems, performance evaluation, pointer swizzling

2 The integration of virtual memory management and interprocess communication in Accent
Robert Fitzgerald, Richard F. Rashid
May 1986 **ACM Transactions on Computer Systems (TOCS)**, Volume 4 Issue 2
Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, citings, index terms

The integration of virtual memory management and interprocess communication in the Accent network operating system kernel is examined. The design and implementation of the Accent memory management system is discussed and its performance, both on a series of message-oriented benchmarks and in normal operation, is analyzed in detail.

3 Virtual memory on a narrow machine for an object-oriented language
Ted Kaehler
June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 21 Issue 11
Full text available: pdf(1.66 MB) Additional Information: full citation, abstract, references, citings, index terms

LOOM (Large Object-Oriented Memory) is a virtual memory implemented in software that supports the Smalltalk-80(™) programming language and environment on the Xerox

Dorado computer. LOOM provides 8 billion bytes of secondary memory address space and is specifically designed to run on computers with a narrow word size (16-bit wide words). All storage is viewed as objects that contain fields. Objects may have an average size as small as 10 fields. LOOM swaps objects between primary and s ...

4 CONS should not CONS its arguments, or, a lazy alloc is a smart alloc

Henry G. Baker

March 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 3

Full text available: pdf(1.52 MB) Additional Information: full citation, abstract, index terms

Lazy allocation is a model for allocating objects on the execution stack of a high-level language which does not create dangling references. Our model provides safe transportation into the heap for objects that may survive the deallocation of the surrounding stack frame. Space for objects that do not survive the deallocation of the surrounding stack frame is reclaimed without additional effort when the stack is popped. Lazy allocation thus performs a first-level garbage collection, and if ...

5 Special system-oriented section: the best of SIGMOD '94: QuickStore: a high performance mapped object store

Seth J. White, David J. DeWitt

October 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 4 Issue 4

Full text available: pdf(2.58 MB) Additional Information: full citation, abstract, references, citings

QuickStore is a memory-mapped storage system for persistent C++, built on top of the EXODUS Storage Manager. QuickStore provides fast access to in-memory objects by allowing application programs to access objects via normal virtual memory pointers. This article presents the results of a detailed performance study using the OO7 benchmark. The study compares the performance of QuickStore with the latest implementation of the E programming language. The QuickStore and E systems exemplify the two ba ...

Keywords: benchmark, client-server, memory-mapped, object-oriented, performance, pointer swizzling

6 Supporting dynamic data structures on distributed-memory machines

Anne Rogers, Martin C. Carlisle, John H. Reppy, Laurie J. Hendren

March 1995 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 17 Issue 2

Full text available: pdf(2.05 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Compiling for distributed-memory machines has been a very active research area in recent years. Much of this work has concentrated on programs that use arrays as their primary data structures. To date, little work has been done to address the problem of supporting programs that use pointer-based dynamic data structures. The techniques developed for supporting SPMD execution of array-based programs rely on the fact that arrays are statically defined and directly addressable. Recursive data s ...

Keywords: dynamic data structures

7 CRegs: a new kind of memory for referencing arrays and pointers

H. Dietz, C. H. Chi

November 1988 **Proceedings of the 1988 ACM/IEEE conference on Supercomputing**

Full text available: pdf(848.04 KB) Additional Information: full citation, abstract, references, citings, index

terms

Often, pointer and subscripted array references touch memory locations for which there are several possible aliases, hence these references cannot be made from registers. Although conventional caches can increase performance somewhat, they do not provide many of the benefits of registers, and do not permit the compiler to perform many optimizations associated with register references. The CReg (pronounced "C-Reg") mechanism combines the hardware structures of cache and registers ...

8 QuickStore: a high performance mapped object store

Seth J. White, David J. DeWitt

May 1994 **ACM SIGMOD Record , Proceedings of the 1994 ACM SIGMOD international conference on Management of data**, Volume 23 Issue 2

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents, QuickStore, a memory-mapped storage system for persistent C++ built on top of the EXODUS Storage Manager. QuickStore provides fast access to in-memory objects by allowing application programs to access objects via normal virtual memory pointers. The paper also presents the results of a detailed performance study using the OO7 benchmark. The study compares the performance of QuickStore with the latest implementation of the E programming language. These systems exemplify ...

9 A real-time garbage collector based on the lifetimes of objects

Henry Lieberman, Carl Hewitt

June 1983 **Communications of the ACM**, Volume 26 Issue 6

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

In previous heap storage systems, the cost of creating objects and garbage collection is independent of the lifetime of the object. Since objects with short lifetimes account for a large portion of storage use, it is worth optimizing a garbage collector to reclaim storage for these objects more quickly. The garbage collector should spend proportionately less effort reclaiming objects with longer lifetimes. We present a garbage collection algorithm that (1) makes storage for short-li ...

Keywords: LISP, algorithms, languages, lisp, object-oriented programming, parallel processing, performance, real-time garbage collection, reference counting, virtual memory

10 Lightweight shared objects in a 64-bit operating system

Jeffrey S. Chase, Henry M. Levy, Edward D. Lazowska, Miche Baker-Harvey

October 1992 **ACM SIGPLAN Notices , conference proceedings on Object-oriented programming systems, languages, and applications**, Volume 27 Issue 10

Full text available: pdf(2.08 MB) Additional Information: full citation, references, citings, index terms

11 MULTILISP: a language for concurrent symbolic computation

Robert H. Halstead

October 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 7 Issue 4

Full text available: pdf(3.30 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Multilisp is a version of the Lisp dialect Scheme extended with constructs for parallel execution. Like Scheme, Multilisp is oriented toward symbolic computation. Unlike some parallel programming languages, Multilisp incorporates constructs for causing side effects

and for explicitly introducing parallelism. The potential complexity of dealing with side effects in a parallel context is mitigated by the nature of the parallelism constructs and by support for abstract data types: a recommende ...

12 Reducing the latency of a real-time garbage collector

Ralph E. Johnson

March 1992 **ACM Letters on Programming Languages and Systems (LOPLAS)**, Volume 1 Issue 1

Full text available: pdf(905.05 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper shows how to make the latency of scanning a page in the Appel-Ellis-Li real-time garbage collector be proportional only to the number of object references on a page (the page size), instead of to the sum of the sizes of the objects referenced by the page. This makes the garbage collection algorithm much more suitable for real-time systems.

Keywords: garbage collection

13 Query evaluation techniques for large databases

Goetz Graefe

June 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 2

Full text available: pdf(9.37 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Database management systems will continue to manage large data volumes. Thus, efficient algorithms for accessing and manipulating large sets and sequences will be required to provide acceptable performance. The advent of object-oriented and extensible database systems will not solve this problem. On the contrary, modern data models exacerbate the problem: In order to manipulate large sets of complex objects as efficiently as today's database systems manipulate simple records, query-processi ...

Keywords: complex query evaluation plans, dynamic query evaluation plans, extensible database systems, iterators, object-oriented database systems, operator model of parallelization, parallel algorithms, relational database systems, set-matching algorithms, sort-hash duality

14 Efficient logic variables for distributed computing

Seif Haridi, Peter Van Roy, Per Brand, Michael Mehl, Ralf Scheidhauer, Gert Smolka

May 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 3

Full text available: pdf(572.35 KB) Additional Information: full citation, abstract, references, citings, index terms

We define a practical algorithm for distrubuted rational tree unification and prove its correctness in both the off-line and on-line cases. We derive the distributed algorithm from a centralized one, showing clearly the trade-offs between local and distributed execution. The algorithm is used to realize logic variables in the Mozart Programming System, which implements the Oz language (see http://www/mozart-oz.org). Oz appears to the programmer as a concurrent object-oriented language with ...

Keywords: Mozart, Oz, distributed algorithms

15 An evaluation of automatic object inline allocation techniques

Julian Dolby, Andrew A. Chien

October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 33 Issue 10

Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented languages such as Java and Smalltalk provide a uniform object reference model, allowing objects to be conveniently shared. If implemented directly, these uniform reference models can suffer in efficiency due to additional memory dereferences and memory management operations. Automatic *inline allocation* of child objects within parent objects can reduce overheads of heap-allocated pointer-referenced objects.We present three compiler analyses to identify inlinable fields by t ...

16 HAC: hybrid adaptive caching for distributed storage systems

Miguel Castro, Atul Adya, Barbara Liskov, Andrew C. Meyers

October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5

Full text available: pdf(2.21 MB) Additional Information: full citation, references, citings, index terms

17 Special issue on persistent object systems: Orthogonally persistent object systems

Malcolm Atkinson, Ronald Morrison

July 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 4 Issue 3

Full text available: pdf(5.02 MB) Additional Information: full citation, abstract, references, citings

Persistent Application Systems (PASs) are of increasing social and economic importance. They have the potential to be long-lived, concurrently accessed, and consist of large bodies of data and programs. Typical examples of PASs are CAD/CAM systems, office automation, CASE tools, software engineering environments, and patient-care support systems in hospitals. Orthogonally persistent object systems are intended to provide improved support for the design, construction, maintenance, and operation o ...

Keywords: database programming languages, orthogonal persistence, persistent application systems, persistent programming languages

18 Static and dynamic partitioning of pointers as links and threads

David S. Wise, Joshua Walgenbach

June 1996 **ACM SIGPLAN Notices , Proceedings of the first ACM SIGPLAN international conference on Functional programming**, Volume 31 Issue 6

Full text available: pdf(921.56 KB) Additional Information: full citation, abstract, references, citings, index terms

Identifying some pointers as invisible threads, for the purposes of storage management, is a generalization from several widely used programming conventions, like threaded trees. The necessary invariant is that nodes that are accessible (without threads) emit threads only to other accessible nodes. Dynamic tagging or static typing of threads ameliorates storage recycling both in functional and imperative languages.We have seen the distinction between threads and links sharpen both hardware- and ...

Keywords: garbage collection, reference counting, storage management, tags

19 Hardware and software support for efficient exception handling

Chandramohan A. Thekkath, Henry M. Levy

November 1994

Proceedings of the sixth international conference on Architectural support for programming languages and operating systems, Volume 29 , 28 Issue 11 , 5

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Program-synchronous exceptions, for example, breakpoints, watchpoints, illegal opcodes, and memory access violations, provide information about exceptional conditions, interrupting the program and vectoring to an operating system handler. Over the last decade, however, programs and run-time systems have increasingly employed these mechanisms as a performance optimization to detect normal and expected conditions. Unfortunately, current archi ...

20 Mobile objects in distributed Oz

Peter Van Roy, Seif Haridi, Per Brand, Gert Smolka, Michael Mehl, Ralf Scheidhauer
September 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 5

Full text available: pdf(484.83 KB) Additional Information: full citation, abstract, references, citings, index terms

Some of the most difficult questions to answer when designing a distributed application are related to mobility: what information to transfer between sites and when and how to transfer it. Network-transparent distribution, the property that a program's behavior is independent of how it is partitioned among sites, does not directly address these questions. Therefore we propose to extend all language entities with a network behavior that enables efficient distributed programm ...

Keywords: latency tolerance, mobile objects, network transparency

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Published before July 1999
Terms used **access object pointer reference memory**

Found **4,184** of **98,628**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Special issue on persistent object systems: Adaptable pointer swizzling strategies in object bases: design, realization, and quantitative analysis

Alfons Kemper, Donald Kossmann

July 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 4 Issue 3

Full text available: pdf(2.69 MB) Additional Information: full citation, abstract, references, citings

In this article, different techniques for *"pointer swizzling"* are classified and evaluated for optimizing the access to main-memory resident persistent objects. To speed up the access along inter-object references, the persistent pointers in the form of unique object identifiers (OIDs) are transformed (swizzled) into main-memory pointers (addresses). Pointer swizzling techniques can be divided into two classes: (1) those that allow replacement of swizzled objects from the buffer before th ...

Keywords: object-oriented database systems, performance evaluation, pointer swizzling

2 Efficient detection of all pointer and array access errors

Todd M. Austin, Scott E. Breach, Gurindar S. Sohi

June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation**, Volume 29 Issue 6

Full text available: pdf(1.62 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a pointer and array access checking technique that provides complete error coverage through a simple set of program transformations. Our technique, based on an extended safe pointer representation, has a number of novel aspects. Foremost, it is the first technique that detects all spatial and temporal access errors. Its use is not limited by the expressiveness of the language; that is, it can be applied successfully to compiled or interpreted languages with subscripted and mutabl ...

3 Static grouping of small objects to enhance performance of a paged virtual memory

James W. Stamos

May 1984 **ACM Transactions on Computer Systems (TOCS)**, Volume 2 Issue 2

Full text available: pdf(1.79 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: Smalltalk, initial placement, object-oriented, paging, programing restructuring, reference trace compression, static grouping, virtual memory

4 Virtual memory on a narrow machine for an object-oriented language

Ted Kaehler

June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 21 Issue 11

Full text available: pdf(1.66 MB) Additional Information: full citation, abstract, references, citings, index terms

LOOM (Large Object-Oriented Memory) is a virtual memory implemented in software that supports the Smalltalk-80(™) programming language and environment on the Xerox Dorado computer. LOOM provides 8 billion bytes of secondary memory address space and is specifically designed to run on computers with a narrow word size (16-bit wide words). All storage is viewed as objects that contain fields. Objects may have an average size as small as 10 fields. LOOM swaps objects between primary and s ...

5 Protection traps and alternatives for memory management of an object-oriented language

Antony L. Hosking, J. Eliot B. Moss

December 1993 **ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles**, Volume 27 Issue 5

Full text available: pdf(1.48 MB) Additional Information: full citation, abstract, references, citings, index terms

Many operating systems allow user programs to specify the protection level (inaccessible, read-only, read-write) of pages in their virtual memory address space, and to handle any protection violations that may occur. Such page-protection techniques have been exploited by several user-level algorithms for applications including generational garbage collection and persistent stores. Unfortunately, modern hardware has made efficient handling of page protection faults more difficult. Moreover, page- ...

6 Special system-oriented section: the best of SIGMOD '94: QuickStore: a high performance mapped object store

Seth J. White, David J. DeWitt

October 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 4 Issue 4

Full text available: pdf(2.58 MB) Additional Information: full citation, abstract, references, citings

QuickStore is a memory-mapped storage system for persistent C++, built on top of the EXODUS Storage Manager. QuickStore provides fast access to in-memory objects by allowing application programs to access objects via normal virtual memory pointers. This article presents the results of a detailed performance study using the OO7 benchmark. The study compares the performance of QuickStore with the latest implementation of the E programming language. The QuickStore and E systems exemplify the two ba ...

Keywords: benchmark, client-server, memory-mapped, object-oriented, performance, pointer swizzling

7 A real-time garbage collector based on the lifetimes of objects

Henry Lieberman, Carl Hewitt

June 1983 **Communications of the ACM**, Volume 26 Issue 6

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

In previous heap storage systems, the cost of creating objects and garbage collection is independent of the lifetime of the object. Since objects with short lifetimes account for a large portion of storage use, it is worth optimizing a garbage collector to reclaim storage for these objects more quickly. The garbage collector should spend proportionately less effort reclaiming objects with longer lifetimes. We present a garbage collection algorithm that (1) makes storage for short-li ...

Keywords: LISP, algorithms, languages, lisp, object-oriented programming, parallel processing, performance, real-time garbage collection, reference counting, virtual memory

8 Storage reclamation in object oriented database systems

Margaret H. Butler

December 1987 **ACM SIGMOD Record , Proceedings of the 1987 ACM SIGMOD international conference on Management of data**, Volume 16 Issue 3

Full text available: pdf(1.46 MB) Additional Information: full citation, abstract, references, citings, index terms

When providing data management for nontraditional data, database systems encounter storage reclamation problems similar to those encountered by virtual memory managers. The paging behavior of existing automatic storage reclamation schemes as applied to objects stored in a database management system is one indicator of the performance cost of various features of storage reclamation algorithms. The results of modeling the paging behavior suggest that Mark and Sweep causes many more input/outp ...

9 Using the co-existence approach to achieve combined functionality of object-oriented and relational systems

R. Ananthanarayanan, V. Gottemukkala, W. Kaefer, T. J. Lehman, H. Pirahesh

June 1993 **ACM SIGMOD Record , Proceedings of the 1993 ACM SIGMOD international conference on Management of data**, Volume 22 Issue 2

Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, index terms

Once considered a novelty, object oriented systems have now entered the mainstream. Their impressive performance and rich type systems have created a demand for object oriented features in other areas, such as relational database systems. We believe the current efforts to combine object oriented and relational features into a single hybrid system will fall short of the mark, whereas our approach, the co-existence approach, has the distinction of requiring far less work, but ...

10 QuickStore: a high performance mapped object store

Seth J. White, David J. DeWitt

May 1994 **ACM SIGMOD Record , Proceedings of the 1994 ACM SIGMOD international conference on Management of data**, Volume 23 Issue 2

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents, QuickStore, a memory-mapped storage system for persistent C++ built on top of the EXODUS Storage Manager. QuickStore provides fast access to in-memory objects by allowing application programs to access objects via normal virtual memory pointers. The paper also presents the results of a detailed performance study using the OO7 benchmark. The study compares the performance of QuickStore with the latest implementation of the E programming language. These systems exemplify ...

11 Maps: a compiler-managed memory system for raw machines

Rajeev Barua, Walter Lee, Saman Amarasinghe, Anant Agarwal

May 1999 **ACM SIGARCH Computer Architecture News , Proceedings of the 26th annual international symposium on Computer architecture**, Volume 27 Issue 2

Full text available: pdf(231.60 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

This paper describes Maps, a compiler managed memory system for Raw architectures. Traditional processors for sequential programs maintain the abstraction of a unified memory by using a single centralized memory system. This implementation leads to the infamous "Von Neumann bottleneck," with machine performance limited by the large memory latency and limited memory bandwidth. A Raw architecture addresses this problem by taking advantage of the rapidly increasing transistor budget to move much of ...

12 A shared, segmented memory system for an object-oriented database

Mark F. Hornick, Stanley B. Zdonik

January 1987 **ACM Transactions on Information Systems (TOIS)**, Volume 5 Issue 1

Full text available: pdf(2.05 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper describes the basic data model of an object-oriented database and the basic architecture of the system implementing it. In particular, a secondary storage segmentation scheme and a transaction-processing scheme are discussed. The segmentation scheme allows for arbitrary clustering of objects, including duplicates. The transaction scheme allows for many different sharing protocols ranging from those that enforce serializability to those that are nonserializable and require communi ...

13 Design of the Mneme persistent object store

J. Eliot B. Moss

April 1990 **ACM Transactions on Information Systems (TOIS)**, Volume 8 Issue 2

Full text available: pdf(3.22 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The Mneme project is an investigation of techniques for integrating programming language and database features to provide better support for cooperative, information-intensive tasks such as computer-aided software engineering. The project strategy is to implement efficient, distributed, persistent programming languages. We report here on the Mneme persistent object store, a fundamental component of the project, discussing its design and initial prototype. Mneme stores objects

14 Memory forwarding: enabling aggressive layout optimizations by guaranteeing the safety of data relocation

Chi-Keung Luk, Todd C. Mowry

May 1999 **ACM SIGARCH Computer Architecture News , Proceedings of the 26th annual international symposium on Computer architecture**, Volume 27 Issue 2

Full text available: pdf(196.77 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

By optimizing data layout at run-time, we can potentially enhance the performance of caches by actively creating spatial locality, facilitating prefetching, and avoiding cache conflicts and false sharing. Unfortunately, it is extremely difficult to guarantee that such optimizations are *safe* in practice on today's machines, since accurately updating *all* pointers to an object requires perfect alias information, which is well beyond the scope of the compiler for languages such as C. T ...

15 The integration of virtual memory management and interprocess communication in Accent

Robert Fitzgerald, Richard F. Rashid

May 1986 **ACM Transactions on Computer Systems (TOCS)**, Volume 4 Issue 2

Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, citings, index

terms

The integration of virtual memory management and interprocess communication in the Accent network operating system kernel is examined. The design and implementation of the Accent memory management system is discussed and its performance, both on a series of message-oriented benchmarks and in normal operation, is analyzed in detail.

16 Lightweight shared objects in a 64-bit operating system
Jeffrey S. Chase, Henry M. Levy, Edward D. Lazowska, Miche Baker-Harvey
October 1992 **ACM SIGPLAN Notices , conference proceedings on Object-oriented programming systems, languages, and applications**, Volume 27 Issue 10
Full text available: pdf(2.08 MB) Additional Information: full citation, references, citings, index terms

17 Improving locality of reference in a garbage-collecting memory management system
Robert Courts
August 1988 **Communications of the ACM**, Volume 31 Issue 9
Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, citings, index terms

Modern Lisp systems make heavy use of a garbage-collecting style of memory management. Generally, the locality of reference in garbage-collected systems has been very poor. In virtual memory systems, this poor locality of reference generally causes a large amount of wasted time waiting on page faults or uses excessively large amounts of main memory. An adaptive memory management algorithm, described in this article, allows substantial improvement in locality of reference. Performance measur ...

18 CRegs: a new kind of memory for referencing arrays and pointers
H. Dietz, C. H. Chi
November 1988 **Proceedings of the 1988 ACM/IEEE conference on Supercomputing**
Full text available: pdf(848.04 KB) Additional Information: full citation, abstract, references, citings, index terms

Often, pointer and subscripted array references touch memory locations for which there are several possible aliases, hence these references cannot be made from registers. Although conventional caches can increase performance somewhat, they do not provide many of the benefits of registers, and do not permit the compiler to perform many optimizations associated with register references. The CReg (pronounced "C-Reg") mechanism combines the hardware structures of cache and registers ...

19 A performance evaluation of pointer-based joins
Eugene J. Shekita, Michael J. Carey
May 1990 **ACM SIGMOD Record , Proceedings of the 1990 ACM SIGMOD international conference on Management of data**, Volume 19 Issue 2
Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings, index terms

In this paper we describe three pointer-based join algorithms that are simple variants of the nested-loops, sort-merge, and hybrid-hash join algorithms used in relational database systems. Each join algorithm is described and an analysis is carried out to compare the performance of the pointer-based algorithms to their standard, non-pointer-based counterparts. The results of the analysis show that the pointer-based algorithms can provide significant performance gains in many situations. The ...

20 Object fault handling for persistent programming languages: a performance evaluation
Antony L. Hosking, J. Eliot B. Moss
October 1993 **ACM SIGPLAN Notices , Proceedings of the eighth annual conference on**

Object-oriented programming systems, languages, and applications,
Volume 28 Issue 10
Full text available: pdf(1.64 MB) Additional Information: full citation, references, citings, index terms

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Google Web Images Groups News Froogle Local[New!] more »
"session memory" "call memory" Search Advanced Search Preferences

Web Results **1** - **10** of about **33** for **"session memory" "call memory"**. (**0.25** seconds)

Glossary
... **Call Memory**. The memory that the memory manager uses to allocate new objects ... **Session Memory**. The memory that the memory manager uses to hold objects ...
www.stanford.edu/dept/itss/docs/ oracle/9i/java.920/a96656/glossary.htm - 16k - Cached - Similar pages

Java Platform for the Enterprise
... For example, new objects are allocated in fast and cheap **call memory**, ... while it uses a copying collector for **session memory** where compactness of ...
lbd.epfl.ch/f/teaching/courses/ oracle8i/java.815/a64682/02entapp.htm - 29k - Cached - Similar pages

Oracle9i Java Application Performance
... At the end of each call, the Oracle9i JVM copies the **call memory** that is ... the static variables in each class into **session memory** so that it can be ...
www.lc.leidenuniv.nl/awcourse/ oracle/java.920/a96656/perf.htm - 127k - Cached - Similar pages

Writing Java Applications on Oracle9i
... Garbage collection, **session memory**, and **call memory** exist solely for each ... **session memory** for static variables and across **call memory** needs; **call** ...
www.lc.leidenuniv.nl/awcourse/ oracle/java.920/a96656/appover.htm - 129k - Cached - Similar pages

[PDF] Oracle8i Release 3 New Feature Summary
File Format: PDF/Adobe Acrobat - View as HTML
... **session memory**, which exists for the duration of the session (a connection, state and execution. context). -. permanent or global memory, ...
www.oracle.com/technology/ products/oracle8i/pdf/8iR3_nfs.pdf - Similar pages

[PDF] Oracle8
File Format: PDF/Adobe Acrobat - View as HTML
... of memory being used help to keep call and **session memory** usage in ... example, Aurora uses generational scavenging in **call memory** where it typically ...
www.oracle.com/technology/ docs/tech/java/jsp/pdf/concepts.pdf - Similar pages

[PDF] Delivering the Promise of Internet Computing: Integrating Java ...
File Format: PDF/Adobe Acrobat - View as HTML
... **Call Memory**. **Session Memory** Shared Memory. JDK libraries. 8i libraries ... **Session Memory**. **Call Memory**. New Space. Old Space. Survived a Call ...
database.sarang.net/database/ oracle/java/delivering_the_promise_twp.pdf - Similar pages

Glossary
... **Call Memory**. The memory that the memory manager uses to allocate new objects. CLASSPATH. The environment variable (or command line argument) that the ...
database.in2p3.fr/.../ java.101/b12021/glos.htm - 23k - Cached - Similar pages

[PDF] Oracle9
File Format: PDF/Adobe Acrobat - View as HTML
... **session memory** for static variables and across **call memory** needs. s. **call memory** for variables that exist within a single call. Session Lifetime ...
www.computing.dcu.ie/~mark/ TEACHING/ORACLE/JavaDevGuide.pdf - Similar pages

[PDF] Mobile agents in a distributed heterogeneous database system ...
File Format: PDF/Adobe Acrobat
... JVM (40 KB), a **session memory**, which is used for static. variables, and a **call memory** for the instance variables of. the currently executed method. ...
ieeexplore.ieee.org/iel5/ 7804/21448/00994247.pdf?arnumber=994247 - Similar pages

Gooogle ▶
Result Page: 1 2 3 **Next**

"session memory" "call memory" Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google